PAC Continuous State Online Multitask Reinforcement Learning with Identification

Yao Liu
Peking University
Beijing, PR China
liuyao@pku.edu.cn

Zhanhan Guo
Carnegie Mellon University
Pittsburgh, PA, United States
zguo@cs.cmu.edu

Emma Brunskill
Carnegie Mellon University
Pittsburgh, PA, United States
ebrun@cs.cmu.edu

ABSTRACT

One key feature of a general intelligent autonomous agent is to be able to learn from past experience to improve future performance. In this paper we consider how an agent can leverage prior experience from performing reinforcement learning in order to learn faster in future tasks. We introduce the first, to our knowledge, probably approximately correct (PAC) RL algorithm COMRLI for sequential multi-task learning across a series of continuous-state, discrete-action RL tasks. We assume tasks are sampled from a finite number of clusters of Markov decision processes, and provide a bound on the number of steps on which the algorithm makes a suboptimal decision that is substantially smaller on later tasks. We also provide preliminary evidence to suggest our approach may be useful in practice, by showing encouraging simulation performance in a standard domain where it compares favorably to a state-of-the-art algorithm.

General Terms

Algorithms

1. INTRODUCTION

A key feature of an intelligent, autonomous agent is to be able to learn from past experience to improve future performance. In many applications, including robotics, consumer marketing, and healthcare, such an agent will be performing a series of reinforcement learning (RL) tasks modeled as Markov Decision Processes (MDPs) with a continuous state space and a discrete action space. In this paper we are interested in formally quantifying the amount of experience needed for the agent to learn to make good decisions while learning from past experience in a series of such tasks.

Our work falls into the broader class of transfer, lifelong and multi-task sequential decision making. There has been significant interest in this area for the past two decades, and novel algorithms have been introduced with promising empirical performance. Transfer learning has typically focused on leveraging experience from a single source task to improve performance on a single target task. Lifelong learning typically refers to doing a series of tasks and improving performance on later tasks. Multi-task learning often refers to having experience from a set of distinct tasks, and using that experience to improve performance on a new target task. This can be done in either a batch setting, or an online setting where the new target task is added to the set of source tasks to accelerate performance on later tasks. This final setting is the one we consider in this paper. Note that the topic of curriculum learning, where an agent may choose the order in which to complete tasks, is a very interesting question that is outside the scope of our current work: here we assume that the stochastic environment selects the next task to provide to an agent.

Encouraging recent work has shown substantially improved empirical performance on online multi-task continuous state RL [1, 4]. However, there has been very little theoretical analysis of this setup. Exceptions include work by Lazaric et al. [12] that shows a bound on the estimated value function used by transferring samples from prior tasks, but they consider the batch setting (where one has a set of samples from a target task) and do not handle online exploration/exploitation. Very recent work by Eaton et al. [2] uses a policy search method and provides regret bounds, but the regret bounds are with respect to the best policy in their policy class with additional structural assumptions, rather than the true optimal policy. To our knowledge there has been no work on bounding the amount of experience needed to make good decisions in later continuous-state RL tasks by leveraging prior experience in tasks with the same state and action space, but different dynamics and/or reward models.

In this paper we help to fill this gap by introducing a PAC RL algorithm (Continuous-State Online Multitask RL with Identification a.k.a COMRLI) for online multi-task learning across a series of continuous-state, discrete-action RL tasks with the same state and action spaces. We assume that each task is drawn from a stationary distribution over C MDP clusters: though for simplicity most of our analysis will assume that there is a single MDP within each cluster, our results also extend to when each cluster consists of MDPs with very similar transition and reward models. We prove that the number of steps on which the agent may take a non ϵ-optimal action, known as the sample complexity, will be substantially smaller in later tasks, scaling as a function of C rather than the size (covering number) of the state-action space. Our work builds on recent advances in PAC discrete-state online multi-task RL algorithms [5] and single task PAC continuous-state reinforcement learning [16]. We also provide encouraging, preliminary empirical performance on a standard domain where our algorithm exceeds a state-of-the-art continuous-state multitask RL algorithm.

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

Our algorithm proceeds in two stages. In the first stage, our approach learns a good estimate of the optimal Q-function of every given task (in our setting, after a bounded number of tasks, all C tasks will have been encountered with high probability). In the second stage, we leverage this prior information to speed learning in new tasks by using a novel method to quickly identify the new task. Intuitively we leverage a mixing assumption on the underlying MDPs to enable us to eliminate possible MDPs by checking that we can reach different states in a bounded number of steps with high probability. This identification process, paired with assuming that tasks are sampled from a stationary (not adversarial) distribution, enables our overall sample complexity to scale as a function of the number of tasks rather than the size of the state-action space.

2. RELATED WORK

There has been much research on methods for leveraging past experience. One of the simple batch settings focuses on transferring experience from a single source task to a single target task. In many cases only the reward of the task is assumed to differ; this reflected domains where the transition dynamics were the same but the goals were different. Later work relaxed the assumption to also allow the transition dynamics to differ, and even allowed partial sharing of the state space. The multi-task setting has also been explored under similar structural assumptions. These approaches all have been shown to improve performance empirically [17].

However there has been far less work that have theoretical guarantees on performance; without theory there is no guarantee that leveraging past experience would not end up harming future performance (negative transfer). The lack of theory is not surprising considering formal bounds for single task RL performance have only been recently developed [11, 10]. These bounds are called Probability Approximately Correct (PAC), since they bound the number of steps in which an algorithm performs less than near-optimal (sample complexity) with high probability. The early work studied discrete state spaces, but more recent work has also developed PAC bounds for continuous state spaces [16, 6], which we build upon.

Some recent work do have theoretical bounds on transfer, but only focus on the simple batch setting of transferring experience from a single source task to a single target task. They also include other assumptions such as access to a generative model of the source task, a linear approximation of the value function in [12], or a discrete state space in [14]. These approaches do not consider the exploration-exploitation trade-off in the online RL setting.

We focus on the continuous state space, multi-task, lifelong learning setting, in which an infinite sequence of tasks are presented sequentially to the algorithm. There has been recent work in this setting using policy gradient [1, 4], and even theoretical bounds [2]; however they require all tasks to share an underlying structure in their policies. Using policy gradient also restricts the scope of their policy class, which the optimal policy may not be a part of.

Our approach builds on related work for the discrete state, multi-task setting [5]: identifying whether a task is the same as a previous task is much faster than learning from scratch. However we use our novel method to quickly identify the current task in a continuous state space.

3. PROBLEM SETTING

In our continuous multi-task RL setting, we assume tasks are drawn sequentially and iid over a finite set $\mathcal{C}$ of C tasks (we will mention in section 4 how this can be relaxed to C clusters of tasks). Each task is a Markov Decision Process (MDP) that can be described by a tuple $(S, A, T, R, \gamma, H, b_0)$: S is a continuous set of states; A is a discrete set of actions; $T(s'|s, a)$ is the probability of transitioning to state s' from s after executing action a; $R(s, a) \in [0, 1]$ is the expected reward from executing action a in state s. The initial state is drawn from b_0. An RL algorithm then executes an action a at every time-step t, after which the MDP returns some reward r_t according to $R(s, a)$, and transitions to a new state according to $T(s'|s, a)$. At every step, the data sample (s_t, a_t, r_t, s_{t+1}) is collected. A task is finished after H steps have passed. The objective of the RL algorithm is to maximize the accumulated discounted rewards i.e. to maximize the value function $V(s) = \mathbb{E}[\sum_{t=1}^{H} \gamma^{t-1} r_t | s_0 = s]$; $\gamma \in [0, 1]$ is the discounted factor, which determines the significance of future rewards. The optimal value function is denoted as $V^*(s)$, and $Q(s, a)$ is the optimal state-action value of starting with action a in state s and then following the optimal policy. Q_{max} is the maximium Q-value in a domain and $Q_{max} \leq \frac{1}{1-\gamma}$.

We assume all tasks have the same state-action space $S \times A$, discount factor γ, and task length H; but, the parameters (T, R, b_0) can differ. The overall objective of the algorithm is to maximize the value function $V(s)$ for all tasks.

While we acknowledge that our assumptions are not suitable for all situations, we believe our setting captures a general set of important domains such as user modeling where different groups of users may have similar behavior (see examples from [13] or [15]).

We prove a high probability, polynomial bound on the sample complexity of our algorithm (PAC bound). The sample complexity is the total number of steps on which the expected value of algorithm is less than ϵ-optimal i.e. formally, the number of steps t where $V(s_t) \leq V^*(s_t) - \epsilon$ for all tasks.

4. ALGORITHM

Intuitively, identifying which of C possible tasks a new task is should require much less experience than learning the parameters of a new task from scratch. Once a task's identity is known, we can leverage all prior knowledge about this task, and immediately start using a good policy.

We use the optimal Q-function instead of the model parameters as the evidence to distinguish different tasks; however we still assume that different tasks have different models, so we only cluster tasks that have the same model. As we will discuss later on, the main reason for using Q-functions rather than model parameters like in prior work is due to the continuous state space.

We divide the sequence of tasks into two phases. In phase 1, in each task, we try to explore the whole state space efficiently to learn a good estimate of the optimal Q-function of every task over the entire state space. At the end of phase 1, we cluster the tasks according to the estimated Q-functions. Since tasks are sampled iid from C possible tasks, we can set the length of phase 1 (see Lemma 2) so that, with high probability, all C types are encountered. In phase 2, at the beginning of each task we try to identify which of the C task clusters it belongs to. After identifying it, we run the single

Algorithm 1 Continuous State Online Multitask RL with Identification (COMRLI)

Require: $T_1, \bar{C}, L_Q, \epsilon, \Gamma, H$
  for $t = 1, 2, \ldots T_1$ **do**
    Receive an unknown MDP $M_t \in \mathcal{M}$
    Run algorithm 2 on M_t with (Γ, D)-*known*.
    For all remaining steps until H steps, execute C-PACE algorithm on M_t
    Store all samples as a set $Sample_{M_t}$
  Cluster all tasks into $\widehat{C} \leq \bar{C}$ groups and combine their sample sets.
  for $t = T_1 + 1, \ldots T$ **do**
    Receive unknown MDP $M_t \in \mathcal{M}$
    Run algorithm 3 on M_t
    if M_t is identified **then**
      Combine samples from M_t to the group

task C-PACE algorithm using the all the previous samples collected from all the tasks within that cluster. Phase 2 is used for all subsequent tasks. We will shortly prove that this approach allows us to improve the sample complexity without incurring negative transfer in terms of our theoretical bounds.

4.1 Assumptions and definitions

Before we illustrate details of our algorithm, we need to introduce some assumptions:

1. There exists a distance metric $d[\cdot, \cdot]$ (e.g. Euclidean in our experiments) in which the optimal Q-function $Q_{M_i}(s, a)$ is Lipschitz continuous over (s, a), for any M_i in $\mathcal{C}$.

2. Tasks are sampled from a finite set of C distinct MDPs, and $\bar{C}$ is a known upper bound on C.

3. The optimal Q function for each of the C MDPs must differ in at least one state-action pair, e.g. for any two MDPs M_i and M_j, there must exist at least one (s, a) pair such that $||Q_i(s, a) - Q_j(s, a)||_2 \geq \Gamma > 0$.

4. There is a finite diameter $D(\epsilon)$: any (s, a) pair's neighborhood (radius at most ϵ) is reachable from any other (s, a) pair in at most an expected $D(\epsilon)$ steps.

The first assumption is a smoothness property over the Q-functions for any MDP. Intuitively, this ensures that close-by state-action pairs can only differ by a bounded amount in their Q-values. This is essential for near optimal learning to be efficient: if any two arbitrarily close state-action pairs can differ an arbitrary amount in their value, then it is necessary to visit all state–action pairs in the space to learn a near-optimal policy; this is an impossible task to do in finite time because the state space is infinite and continuous. Note that the Q-funtion is also Lipschitz continuous over the action space if we use proper a metric (e.g. our experiments). This assumption also implies that one could approximate such an MDP using a finite set of states and actions, because nearby states can only differ a finite amount in their resulting state–action values. This assumption was used and resulted in the tabular representation of the Q-function by a prior paper that introduced a PAC RL algorithm for single task continuous-state MDPs[16].

The second assumption describes our multi-task setting. For simplicity, we assume that there are C distinct MDP clusters with only one MDP in each cluster i.e. only C distinct MDPs, and that each task is one of the MDPs. It is straightforward to relax the assumption to the case where each cluster is made up of MDPs with similar dynamics. [1]

The third assumption says that we require tasks that are from distinct MDPs to differ in their optimal Q-values in at least one state–action pair. This is quite mild: if two tasks do not differ in their optimal Q-values in any state–action pair, then they have identical Q-values and optimal policies, and are likely to have the same model parameters.

The fourth assumption is perhaps the strongest, and represents a restriction on the mixing property of the MDPs considered. A similar assumption has been employed in discrete state–action spaces[3, 5, 8]. Intuitively, it ensures that it is possible to go between two regions of state–action space under some policy in a bounded number of steps. This assumption is needed for our algorithm to perform identification of the current task during phase 2. Fortunately there do exist a number of domains which have bounded diameter. For example, many interesting RL domains are episodic. In episodic settings, the diameter can be no more than twice the episode length for any two states that are reachable within one episode from the starting distribution. Also note that in an episodic domain, only the states that are reachable within one episode from the starting state are relevant to the optimal policy and value function.

Our algorithm also depends on the *Q-gap* between distinct optimal Q-functions.

Definition 1. The Q-gap, Γ, is a positive constant that satisfies that for any two MDP clusters[2] M_i, $M_j \in \mathcal{M}$, there exist some state–action pair (s, a) such that $|Q^*_{M_i}(s, a) - Q^*_{M_j}(s, a)| \geq \Gamma$.

Note that according to the third assumption, the *Q-gap* must always be greater than 0. We do assume our algorithm has an estimate of a lower bound on Γ. In many real world domains, we don't know Γ in advance; in this case the lower bound can be set to $\epsilon/2$, since if the Q-functions between two tasks are closer than $\epsilon/2$, the ϵ-optimal policy for one is still an ϵ-optimal policy for the other and the two tasks would be most likely almost identical anyway. However in most cases, Γ would be much larger than ϵ.

Some more definitions used in algorithm are clarified here:

Definition 2. A state-action is known if it has k visited neighbors, which means $L_Q d[(s, a), (s_i, a_i)] \leq \frac{\epsilon(1-\gamma)}{8}$, where $k = O(\frac{1}{\epsilon^2(1-\gamma)^2})$ according to theorem 3.16 in [16]. This kind of knownness is called as ϵ-known. The knownness in phase 1, (Γ, D)-known, is different and will be discussed later.

[1] If we have C clusters of MDPs, where all MDPs in the same cluster have highly similar transition and reward models, then, our results can be extended; our results immediately apply following 3 if MDPs in the i-th cluster (of C clusters), satisfy the following property: for any two MDP M_1, M_2 in cluster i, for any (s, a) pair,

$$|r_{M_1}(s, a) - r_{M_2}(s, a)| < \frac{\epsilon(1-\gamma)}{16}$$
$$\int_{\mathcal{S}} |T_{M_1}(s'|s, a) - T_{M_2}(s'|s, a)| \mathrm{d}s' < \frac{\epsilon(1-\gamma)^2}{16}$$

[2] Note that different MDP clusters have distinct Q-functions, given our assumption 3 and lemma 3.

Definition 3. For an MDP $M = < S, A, T, R, \gamma, H, b_0 >$, let K be the set of known state-action pairs. The known MDP $M_K = < K \cup sa_{uk}, T, R, \gamma, H, b_0 >$ is defined as follows. sa_{uk} is an additional s-a pair to denote all the unknown s-a pairs. For state-action pairs in K, $r(s,a) = 0$, and for the sa_{uk} state-action, $\mathrm{r}(s,a) = 1$. All the unknown state-action pairs are merged into sa_{uk} and it is an absorbing state-action pair. Transitions to unknown pairs in M are redirected to sa_{uk}, and transitions within K are identical to those defined in the original MDP M.

Definition 4. For a Q-function Q_{M_i}, we use π_{M_i} to denote the greedy policy introduced by Q_{M_i}.

Algorithm 2 Phase 1: Continuous PAC Explore

Require: T_e, L_Q, Γ
  Set the neighborhood radius to $\frac{min\{\Gamma/4, 1/24\}}{L_Q}$
  while some (s,a) is unknown (see Def.7) **do**
    This is a start of new T_e-step episode.
    Find a T_e-step undiscounted optimistic Q-function Q_{0,T_e} by:
    Initialize: $Q_{T_e,T_e}(s,a) = 0$
    for $t = T_e - 1, \ldots, 0$ **do**
      if (s,a) is known **then**
        Find k-NN of (s,a): $(s_j, a_j, r_j, s_j^{'})_{j=1}^{k}$
        $Q_{t,T_e} = \frac{1}{k} \sum_{j=1}^{k} \left(r_j + \max_a Q_{t+1,T_e}\left(s_j^{'}, a\right) + L_Q d_{ij} \right)$
      else
        $Q_{t,T_e} = (T_e - t)$
    Take greedy policy of $Q_{0,T_e}(s,a)$ for next T_e steps.
    if (s,a) is unknown **then**
      Add (s,a,r,s') to the sample set

4.2 Phase 1

In the first phase, we efficiently explore each task to get a good estimate of their optimal Q-functions. We only need the estimated Q-functions to be (Γ, D) known, which means $O(\Gamma)$ (the precise value is set later in the theory section) within the true Q-function, in order to correctly cluster the tasks. Note that Γ may be much bigger than ϵ so phase 1 may not yet give us an ϵ-optimal policy for the tasks. We extend the PAC-EXPLORE algorithm [8] to continuous spaces while maintaining the same guarantees, and use it to explore (see Algorithm 2). To achieve efficient exploration, we construct a new MDP called *known MDP* (see definition 3). Then we find a T_e-step undiscounted optimistic Q-function on this MDP through value iteration, where the notation Q_{t,T_e} means the T_e-step optimistic function in the t^{th} step. The policy introduced by this Q-function will direct exploration towards the less explored (unknown) state-action pairs. After the Q-function is (Γ, D) known, we then execute C-PACE for remaining steps of the task. The k nearest neighbors are defined by the distance used in the Lipschitz property. Note that the number of tasks in Phase 1 is defined in advance, in order to ensure all underlying C tasks are experienced with high probability.

4.2.1 Clustering and Informative State–action Pairs

At the end of each task in phase 1, we solve the fixed-point equations (equation (1) in [16]) in C-PACE to estimate the optimal Q-function of that task given the data gathered. Our analysis in the next section shows that the distance between the estimated Q-functions and the true optimal Q-functions for each task is no more than $\frac{\Gamma}{4}$. This allows our algorithm to cluster together all tasks whose true optimal Q-function differ by no more than a fixed threshold (defined below). After the clustering completes, all the samples within a cluster are merged to form a single sample set. The algorithm then runs the fixed-point method from C-PACE to estimate the optimal Q-function for each cluster. Note that depending on the amount of experience in the cluster, this estimated function may or may not be ϵ-optimal yet.

Once the algorithm computes an estimated optimal Q-function for each cluster, these functions are used to compute a set of informative state–action pairs.

Algorithm 3 Phase 2: Continuous Identify

Require: $\Gamma, \epsilon, \bar{C}, H, T_i, n$
  Initialize version space: $\mathcal{C} \leftarrow \{1, \ldots, \bar{C}\}$
  while $h < H$ **do**
    for $c \in \mathcal{C}$ **do**
      Use algorithm 2 with samples from M_c to find a informative pair $(s,a), s.t. \left| Q_{M_i}^{\pi_i}(s,a) - Q_{M_j}^{\pi_j}(s,a) \right| \geq \frac{7\Gamma}{8}$
      if informative pair (s,a) is reached **then**
        Break the loop.
    for $g = \{i, j\}$ **do**
      for $t = 1 \ldots T_i$ **do** (Monte Carlo estimate)
        Run the greedy policy of Q_{M_g}, π_g, for n steps
        $R_{gt} \leftarrow r(s,a) + \sum_{l=1}^{n} \gamma^l r_l, h \leftarrow h + n$
        $\widetilde{D} \leftarrow 1$
        while Haven't returned to (s,a) **do**
          for $c \in \mathcal{C}$ **do**
            Use M_c to create an informative MDP, and try to go back to (s,a) within $12\widetilde{D} \ln \frac{\bar{C}}{\delta}$ steps.
            if get back to (s,a) **then**
              Break the loop.
          $\widetilde{D} \leftarrow 2\widetilde{D}$
      $R_g \leftarrow \frac{1}{T_i} \sum_{t=1}^{T_i} R_{gt}$
      if $\left| R_g - Q_{M_g}(s,a) \right| \geq \frac{\Gamma}{8}$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \backslash \{g\}$
    if Both i,j haven't been eliminated **then**
      Eliminate the model k with a smaller R_g.
    if Only one group left **then**
      Combine the sample sets and run C-PACE

Definition 5. Given a set of Q-functions, a state-action (s,a) is informative if there exist at least two distinct Q-functions, Q_i and Q_j, such that $|Q_i(s,a) - Q_j(s,a)| \geq \frac{7\Gamma}{8}$.

These informative pairs are the state–action pairs where Q-functions significantly differ, and so distinguish between distinct Q-functions. We will make heavy use of these pairs to identify which cluster new tasks will belong to in phase 2.

4.3 Phase 2

After clustering all the tasks from phase 1, we end up with $\widehat{C}$ clusters, each of which consist of a (merged) sample set and an estimated optimal Q-function. In this phase, we try to identify which cluster each new task belongs to, and once we do we can use that cluster's sample set to jumpstart

learning – we may even immediately get a near-optimal policy if we have enough data in the cluster's sample set.

Identifying the current task requires some care since we only know that the current task has an optimal Q-function that matches one of the $\widehat{C}$ clusters. Of course, if we compute an approximate Q-function in the current task of sufficient accuracy, that is enough to identify which cluster it belongs to; however by the time we have enough data to achieve that accuracy in computing the Q-function, we will only get a little benefit to leveraging prior samples. This will only reduce the exploration needed to being a function of Γ instead of ϵ, but it will not impact the dependence on the size of the state–action space covering number.

Instead, we will rely on informative pairs for identification. For each task in phase 2, we start with a set of $\widehat{C}$ candidate clusters that the new task potentially belongs to. Then we can use their associated Q-functions to compute the set of informative pairs I; we will need at most one informative pair for every pair of distinct clusters. We then repeatedly visit an informative pair and compute an estimate of its Q-value for the current task. Each informative pair distinguishes between two clusters' Q-functions. Then we will compare the estimate to the two optimal Q-functions to determine which cluster should be eliminated as a candidate. Then we move onto another informative pair to eventually eliminate another candidate. This process is repeated until there is only one candidate left. More details follow.

Given an informative set of pairs, we define an informative MDP for the current task:

Definition 6. For an MDP $M = < S \times A, T, R, \gamma >$, let I be the set of informative state-action pairs. The MDP $M_{inform} = < S \times A \setminus I \cup sa_{inform}, T, R, \gamma >$ is defined in a similar way as known MDP in definition 3, replacing the unknown state-action pairs with informative pairs.

To start, we need to quickly reach an informative pair. The main issue with trying to visit informative pairs quickly is that the identity of the new task is unknown. If we knew the identity, then we could create an M_{inform} based on the associated cluster, and execute the resulting policy. Because of our diameter assumption, this policy will quickly reach the target informative pair. But since we actually don't know the identity of the new task, we need to do something else. What we do is try to use the M_{inform} of every candidate cluster. If we fail to reach the target informative pairs, we just move on and try to use M_{inform} of the next candidate cluster. Since we know that one of candidate cluster is the true cluster, we will eventually use its M_{inform}, and successfully reach the target pair. This will require trying to use the M_{inform} of each candidate cluster at most once. With this process, first we can create M_{inform} for each candidate with all the informative pairs I to try to reach any informative pair. After that, we can create M_{inform} for each candidate with a single target informative pair to try to return to that particular informative pair.

Next, we need to use the informative pair to try to eliminate candidates. Note that an informative (s,a) pair is only guaranteed to be informative for a particular pair of candidates (the pair may actually be informative for more than two candidates, but for simplicity we only consider two). After we reach an informative state-action pair, for example where Q_{M_i} and Q_{M_j} is sufficiently different, we run T_i trajectories of π_{M_i} and π_{M_j} from here (using the previous process to return to the informative pair after each trajectory) for an estimate of this task's Q-value at this pair. This is to test which one of these two Q-functions is different with the current new task's Q-function. Since Q_{M_i} and Q_{M_j} are sufficiently different, we know at least one of them will differ with the current new task's Q-value. If the estimate of the Q-value of the current task is not close to the Q-value computed from its candidate, this candidate will be eliminated. If both policies being tested have estimates that are close enough to what they are supposed to be, we will eliminate the candidate with the smaller computed Q-value. Once we eliminate one or both of these candidates, we pick another informative pair belonging to two other candidates to test. Every informative pair leads to eliminating at least one candidate, so we will eventually result in only one candidate left i.e. a successful identification.

After a successful identification it may immediately result in an ϵ-optimal policy if there are enough samples from the cluster; otherwise the data of the current task is collected and merged with the cluster's data. Later on, after encountering tasks from the same cluster over and over again there will eventually be enough data for an ϵ-optimal policy. Thereafter, every new task will get an ϵ-optimal policy immediately after identification. Since identification does not depend on the covering number of the state-action space, the sample complexity of new tasks will then also no longer depend on the size (covering number) of the state-action space. This is the source of the sample efficiency.

4.4 Motivation for Using Q-functions

In prior work [5], the difference between model parameters (the transition and reward dynamics) is used to distinguish between different tasks and to identify a new task. However complications arise if we try the same approach here due to a continuous state space. The transition function becomes a probability density and is hard to represent. We can parameterize it, but that restricts the structure of the transition function. Alternatively, we can use a nonparametric method such as Gaussian Processes (GPs) like in [7], however that introduces extra computational complexities, as opposed to using a simple table to maintain a Q-function. Also, using a GP results in slower learning for single tasks than using the Q-function based method C-PACE [7]. Our approach of using Q-functions is simpler since the representation is just a table, and this same representation is also used for the policy. The only smoothness assumption is on the Q-function, and not on the model parameters.

Using Q-functions rather than model parameters to cluster also opens a new opportunity. Without assuming that different MDPs have different Q-functions, it may be possible that different MDPs end up having the same Q-function and can be clustered together. However we would no longer be able to identify new tasks quickly. Our identification relies on quickly reaching informative states. Computing these policies to reach informative states requires that the dynamics of the new task resemble the dynamics of the cluster it belongs to. If we allowed different MDPs with the same Q-function to be clustered together, we would no longer be able to guarantee that we can reach an informative state quickly.

5. THEORETICAL GUARANTEES

To start the analysis of the sample complexity, we need some additional assumptions and definitions.

1. Each task has at least $p_{min} > 0$ probability to be drawn in phase 1, and phase 1 has at least $\frac{\ln \bar{C}}{p_{min}\delta}$ tasks.

2. The covering number $\mathcal{N}_{\mathcal{SA}}(L_Q, \epsilon)$ is the size of the largest minimal covering set S_c of the state-action space, which means that for any (s, a) there exist k points in S_c, such that for any one of the k points (s_i, a_i): $L_Q d\left[(s, a), (s_i, a_i)\right] \leq \epsilon(1-\gamma)$

3. Given an input δ, all tasks in phase 1 and 2 should be executed for more than H_{min} steps, where:

$$H_{min} = O\left(max\left\{D^2 \ln\left(\frac{\mathcal{N}_{\mathcal{SA}}(L_Q, \Gamma)}{\delta}\right), \frac{Q_{max}^2}{\Gamma^2(1-\gamma)^2} \ln\left(\frac{T_1 \mathcal{N}_{\mathcal{SA}}(L_Q, \Gamma)}{\delta}\right)\right\} \mathcal{N}_{\mathcal{SA}}(L_Q, \Gamma) D\right)$$

The first assumption makes sure we will encounter all distinct tasks in phase 1, and is a common assumption in previous work [5]. The second assumption is to ensure we can sufficiently explore to create good enough estimates of the Q-functions in phase 1. The main PAC bound for our algorithm is described in the theorem below.

THEOREM 1. *For any ϵ and δ, if we run Algorithm 1 for T sequential tasks, each for $H \geq H_{min}$ steps, where H_{min} meets the requirement in the third assumption above. Then the algorithm will select an ϵ-optimal policy on all but at most*

$$O\left(T_1 \max\left\{H_{min}, \min\left\{\zeta_s, H\right\}\right\} + \bar{C}\zeta_s + (T - T_1) \frac{Q_{max}^2 \bar{C}}{\Gamma^2} \ln \frac{\bar{C}}{\delta}\left(\bar{C} D \ln \frac{\bar{C}}{\delta} + \log_\gamma \Gamma\right)\right)$$

steps, with probability $1 - \delta$. ζ_s is the sample complexity of a single task C-PACE algorithm, which is at most $O\left(\frac{Q_{max}^2}{\epsilon^2(1-\gamma)^2} \ln\left(\frac{1}{\epsilon(1-\gamma)}\right) \ln\left(\frac{\mathcal{N}_{\mathcal{SA}}(L_Q, \epsilon)}{\delta}\right) \mathcal{N}_{\mathcal{SA}}(L_Q, \epsilon)\right)$.

T_1 is the number of tasks in phase 1. Before going into the proofs, we are going to examine the improvement in sample complexity of our algorithm over a single-task algorithm.

First, consider the sample complexity of phase 1, which is the first term of the bound. We want to compare H_{min} and ζ_s. If H_{min} is less than or equal to ζ_s, then the single-task complexity ζ_s dominates and we do no worse than the single task case. In most cases, $T_1 < \mathcal{N}_{\mathcal{SA}}(L_Q, \Gamma)$, so H_{min} can be simplified to

$$O\left(max\left\{D^2, \frac{Q_{max}^2}{\Gamma^2(1-\gamma)^2}\right\} \ln\left(\frac{\mathcal{N}_{\mathcal{SA}}(L_Q, \Gamma)}{\delta}\right) D \mathcal{N}_{\mathcal{SA}}(L_Q, \Gamma)\right)$$

Note that the $\mathcal{N}_{\mathcal{SA}}(L_Q, \epsilon)$ term for C-PACE can be much larger than the $\mathcal{N}_{\mathcal{SA}}(L_Q, \Gamma)$ term for H_{min} due to Γ being larger than ϵ. If D is at most $O\left(\frac{1}{\epsilon^{\frac{2}{3}}}\right)$ and $\frac{D}{\Gamma^2}$ is at most $O\left(\frac{1}{\epsilon^2}\right)$, then the sample complexity of phase 1 is no more than C-PACE.

Now we compare the total sample complexity of our algorithm with a single-task algorithm. Even in the worst case, the total sample complexity of our algorithm is about

$$\widetilde{O}\left(\left(\bar{C} + T_1 D\right)\zeta_s + (T - T_1)\frac{\bar{C}^2 D Q_{max}^2}{\Gamma^2}\right)$$

Since in most cases $\bar{C} << \mathcal{N}_{\mathcal{SA}}$ and $\bar{C} + T_1 D$ is constant as T increases, assuming again that $\frac{D}{\Gamma^2}$ is at most $O\left(\frac{1}{\epsilon^2}\right)$ we can get a notable improvement over single task algorithms whose sample complexity is linear with respect to $\mathcal{N}_{\mathcal{SA}}$.

Now we start to prove the theorem. We firstly divide the suboptimal steps of our algorithm into 3 parts, and bound each part. The first part consists of steps in phase 1; the second part consists of steps in phase 2 before identification succeeds; and the third part consists of suboptimal steps after identification succeeds. Besides analyzing the sample complexity, we also need to prove the accuracy of clustering and the correctness of identification. Before that, we introduce some supporting lemmas. Proofs, when omitted, are provided in the supplement.[3]

5.1 Lemmas

PROPOSITION 1. *(lemma 4.5 in [9]) There are at most $k\mathcal{N}_{\mathcal{SA}}(L_Q, \Gamma)$ number of visits to unknown state-action pairs in Algorithm 2, where a known state-action pair means it has k visited neighbors within a distance of $\frac{\Gamma(1-\gamma)}{8L_Q}$.*

PROPOSITION 2. *In algorithm 2, denoting exact Bellman operator by B and the upper bound of Q-value by Q_{max}, if*

$$\frac{4Q_{max}^2}{\epsilon^2} \ln\left(\frac{4\mathcal{N}_{\mathcal{SA}}(L_Q, \Gamma)}{\delta}\right) \leq k \leq \frac{4\mathcal{N}_{\mathcal{SA}}(L_Q, \Gamma)}{\delta}$$

and the radius of the neighborhood is no more than $\frac{\Gamma}{2L_Q}$, then w.p. $1 - \delta/2$, for all known (s, a) (Known is defined in proposition 1), we have for any $t < T_e$:

$$\left|Q_{t,T_e}(s, a) - BQ_{t+1,T_e}(s, a)\right| \leq \Gamma$$

where T_e is the finite horizon length in algorithm 2.

PROPOSITION 3. *Assume $R \in [0, 1]$. Suppose in Algorithm 2, $\left|Q_{t,T_e}(s, a) - BQ_{t+1,T_e}(s, a)\right| \leq \Gamma$, π is a T_e-step greedy policy introduced by Q_{t,T_e}, and Q_{t,T_e}^{π} is the Q value of this policy. Then $\forall$ known (s, a) (Known is defined in proposition 1), $t < T$, $\left|Q_{t,T_e}^*(s, a) - Q_{t,T_e}^{\pi}(s, a)\right| \leq 2(T_e - t)\Gamma$, and*

$$\left|V_{t,T_e}^*(s) \leq V_{t,T_e}^{\pi}(s)\right| \leq 2(T_e - t)\Gamma$$

Before we introduce the following lemmas, we need to introduce a new concept of knownness, (Γ, D)-known.

Definition 7. A state-action pair is (Γ, D)-known when it has at least

$$max\left\{D^2 \ln\left(\frac{\mathcal{N}_{\mathcal{SA}}(L_Q, \Gamma)}{\delta}\right), \frac{Q_{max}^2}{\Gamma^2(1-\gamma)^2} \ln\left(\frac{T_1 \mathcal{N}_{\mathcal{SA}}(L_Q, \Gamma)}{\delta}\right)\right\}$$

visited neighbors within a distance of $\frac{\Gamma(1-\gamma)}{8L_Q}$.

LEMMA 1. *After no more than $O\left(\left(k\mathcal{N}_{SA}(L_Q, \Gamma) + \ln\frac{1}{\delta}\right) D\right)$ steps of Algorithm 2, every state-action pair will have at least k visited neighbors, with probability of $1 - \delta$, where $k \geq k_{min} = O\left(D^2 \ln\left(\frac{\mathcal{N}_{\mathcal{SA}}(L_Q, \Gamma)}{\delta}\right)\right)$, $k \leq k_{max} = O\left(\frac{\mathcal{N}_{\mathcal{SA}}(L_Q, \Gamma)}{\delta}\right)$.*

PROOF. (Sketch) Following the three proposition above, this proof is similar to the proof of Theorem 2 in [8]. The key intuition here is to construct a known MDP and try to achieve an ϵ-optimal policy in this MDP, directing exploration to unknown state-action pairs quickly. □

[3]http://cs.cmu.edu/ ebrun/pac_continuous_multitask_aamas2016.pdf

LEMMA 2. *(lemma 1 in [5]) If* $T_1 \geq \frac{\ln \frac{\bar{C}}{\delta}}{p_{min}}$ *then w.p.* $1-\delta$, *all distinct MDPs will be encountered in phase 1.*

LEMMA 3. *If* M_1 *and* M_2 *are 2 MDPs s.t. for any (s,a),*

$$|r_{M_1}(s,a) - r_{M_2}(s,a)| < \frac{\epsilon(1-\gamma)}{2}$$
$$\int_{\mathcal{S}} |T_{M_1}(s'|s,a) - T_{M_2}(s'|s,a)| \mathrm{d}s' < \frac{\epsilon(1-\gamma)^2}{2}$$

then the optimal Q-functions for M_1 *and* M_2, Q_{M_1} *and* Q_{M_2}, *satisfy that for any (s,a) pair*

$$|Q_{M_1}(s,a) - Q_{M_2}(s,a)| < \epsilon$$

This lemma allows us to relax the assumption of C distinct tasks to C fuzzy clusters. It shows if the tasks within a cluster are similar enough, the Q functions would be viewed as the same, in the sense of ϵ-accuracy.

LEMMA 4. *If all tasks in phase 1 are run for at least* H_{min} *steps, with probability* $1-\delta$, *the following holds:*

1. *For all tasks, any state-action pair is* (Γ, D)*-known.*
2. *Tasks in phase 1 will be clustered correctly w.h.p..*
3. *For any cluster, the max-norm distance between the approximate Q-function and the true optimal Q-function for any task in this cluster is at most* $\frac{5\Gamma}{16}$.

LEMMA 5. *Assume every state-action pair is* (Γ, D)*-known. Then given any start state and desired state-action pair, it is possible to visit the desired state-action pair's neighborhood in no more than* $\widetilde{O}(D)$ *steps with high probability.*

PROOF. Firstly, we construct an MDP M_{inform} such that the desired state-action pairs have unit reward and all others have 0 reward. The desired pair is a self-loop and other transition probabilities are inherited from the true MDP dynamics. We know which point is desired, so we can modify the original samples to become samples in the new MDP M_{inform}. Because now every pair is (Γ, D)-known, so we have $O(D^2)$ samples in every state-action's neighborhood in this M_{inform}. Following a similar analysis of lemma 1, we could find a policy who will reach the desired region within $3D$ steps w.p. $\frac{1}{4}$. Then after $3D \log_{\frac{3}{4}} \delta$ steps the policy could reach the desired region with probability of $1-\delta$. □

LEMMA 6. *When we face an unknown task in phase 2, we could reach any desired state-action pair within* $O(\bar{C}D \ln \frac{\bar{C}}{\delta})$ *steps with probability* $1 - \frac{\delta}{\bar{C}}$.

PROOF. First, consider the case where we know the diameter D. The unknown task must be one of the $\bar{C}$ tasks from phase 1. Our algorithm tries to run each policy in $3D \ln \frac{\bar{C}}{\delta}$ steps to the desired state-action pair using the samples from one of the $\bar{C}$ tasks, thus lemma 6 holds with probability $1 - \frac{\bar{C}}{\delta}$. By trying all policies from the $\bar{C}$ tasks, we will encounter the one policy that corresponds to the same task and reach the desired region with high probability.

If we don't know the diameter, we could use the doubling trick to find an upper bound on D without an increase in the sample complexity. First we try the whole process with $\widetilde{D} = 1$. If we fail, we double the $\widetilde{D}$ and begin a new trial. When $\widetilde{D}$ is bigger than the true value of D, the rest of the analysis is the same as when we know the true diameter. □

LEMMA 7. *If* T_i *in algorithm 3 is at least* $O\left(\frac{Q_{max}^2}{\Gamma^2} \ln \frac{\bar{C}}{\delta}\right)$ *and n is at least* $O(\log_\gamma \Gamma)$, *we could compute an approximate Q value of policy* π *over current task* M': $\widehat{Q}^{\pi_i}_{M'}(s,a) = R_i$ *such that for any* (s,a), $\left|\widehat{Q}^{\pi_i}_{M'}(s,a) - Q^{\pi_i}_{M'}(s,a)\right| \leq \frac{\Gamma}{16}$ *with probability* $1 - \frac{\delta}{\bar{C}}$.

PROOF. (Sketch) The key insight to prove this lemma is that R_i is the empirical mean of discounted rewards ignoring the tail rewards, so we could use concentration measures to bound the difference of R_i with $Q^{\pi_i}_{M'}(s,a)$, which implies T_i should be at least $O\left(\frac{Q_{max}^2}{\Gamma^2} \ln \frac{\bar{C}}{\delta}\right)$. □

LEMMA 8. *If both model* M_i *and* M_j *haven't been eliminated by* $\left|R_g - Q_{M_g}(s,a)\right| \geq \frac{\Gamma}{8}$ *in algorithm 3, then the true model would have a greater* R_g *with high probability.*

PROOF. (Sketch) Without loss of generality, we assume M_i is the true model M'. The condition in the lemma implies $\widehat{Q}^{\pi_i}_{M_i}(s,a)$ is very close to $Q^{\pi_i}_{M_i}(s,a)$ and $\widehat{Q}^{\pi_j}_{M_i}(s,a)$ is very close to $Q^{\pi_j}_{M_j}(s,a)$. Note that π_i is a nearly optimal policy and the difference of $Q^{\pi_i}_{M_i}(s,a)$ with $Q^{\pi_j}_{M_j}(s,a)$ is $O(\Gamma)$ according to the Q-gap assumption. So the $\widehat{Q}^{\pi_j}_{M_i}(s,a)$ would be no more than $\widehat{Q}^{\pi_i}_{M_i}(s,a)$, otherwise it would exceed the optimal Q value. □

LEMMA 9. *After* $O\left(\frac{Q_{max}^2}{\Gamma^2} \bar{C} \ln \frac{\bar{C}}{\delta} \left(\bar{C}D \ln \frac{\bar{C}}{\delta} + \log_\gamma \Gamma\right)\right)$ *steps in phase 2, we could correctly identify the new task w.p.* $1-\delta$.

PROOF. (Sketch) This lemma straightforward follows from lemma 6 and 8. □

5.2 Proof of Main Theorem

PROOF. Recall that we divide the sample complexity into 3 parts. We will show the bound of each part to prove the whole sample complexity.

The first part consists of the steps in phase 1. H_{min} is the lower bound of steps we need to run algorithm 2 in phase 1. After we finish the exploration, we can run C-PACE for remaining steps of the episode $(H - H_{min})$. Note that the unknown state-action pairs of algorithm 2 is actually a subset of the unknown state-action pairs in ϵ-known C-PACE (since $\Gamma \geq \epsilon$ so more states are unknown under the threshold of ϵ-known). So we could view algorithm 2 as part of the initial exploration done in C-PACE. The total sample complexity in phase 1 would be no more than $max\{H_{min}, min(\zeta_s, H)\}$.

The second part consists of the identification during phase 2. Following from lemma 9 we need

$$O\left((T - T_1)\frac{Q_{max}^2 \bar{C}}{\Gamma^2} \ln \frac{\bar{C}}{\delta} \left(\bar{C}D \ln \frac{\bar{C}}{\delta} + \log_\gamma \Gamma\right)\right)$$

samples in $T - T_1$ tasks in phase 2.

The third part is after the identification in phase 2. If the cluster has gathered enough samples we will get an ϵ-optimal policy immediately. If not, we still need to gather more. But since the samples are gathered across all the tasks in one cluster and the number of clusters is at most $\bar{C}$, this only yields an additional sample complexity of $\bar{C}\zeta_s$. Note that to run the C-PACE algorithm, we set all $\epsilon_s, \epsilon_T, \epsilon_d, \epsilon_K$ in C-PACE to $\frac{\epsilon(1-\gamma)}{8}$ so that we could get an ϵ-optimal policy. □

6. EXPERIMENTS

Though our primary contribution is to prove online multi-task RL enables a lower sample complexity in later continuous-state tasks, we also tested its empirical performance on a popular simulated domain, the spring mass damper system.

This domain is characterized by 3 parameters: the spring constant k, the damping constant d and the mass m. The ranges are: $k \in [1, 10]$ $d \in [0.01, 0.2]$ $m \in [0.5, 5]$, which are mirrored from [1]. The state is parameterized by the position and velocity of the mass, and the action is a horizontal force on the mass. Due to the complexity of performing a maximization over a continuous set of actions, we considered a discrete action set of $\{-10, -1, -0.1, 0, 0.1, 1, 10\}$. [4] The transition dynamics are characterized by an ODE system and simulated by the Euler method in our experiment.

The goal is to control the mass starting at $(1, 0)$ to stay in state $(0, 0)$, and the reward is the negative $l2$-norm distance between current state and the goal.

We performed simulation rounds, where each round consists of 50 tasks. Tasks are generated by randomly sampling from three distinct spring-mass MDPs (k,d,m). Each task consists of 100 episodes with each episode having 150 steps: these settings were informed by recent work[1]. We repeat this process for 40 rounds to assess average performance across a series of tasks.

We compare against 3 other baselines. The first is single-task C-PACE: the PAC RL algorithm C-PACE is run for each task independently, yielding no information sharing across tasks. In the second baseline, C-PACE (mixture of all MDPs), is executed across all tasks without distinguishing them. This is a naive form of transfer that treats all tasks as identical. Finally, our third baseline is the state-of-the-art online policy gradient method for multi-task RL, PG-ELLA[1]. PG-ELLA learns a shared basis set across different tasks. Unlike our work, PG-ELLA assumes as input the true identification of each task. PG-ELLA runs 5000 episodes rather than 100 for each task. That's because they need more samples to update policy parameters.

The considered algorithms all have input parameters. For C-PACE (and the parts of our algorithm that use C-PACE), we let $k = 1$ since this domain is deterministic. The Lipschitz constant is set to 0.1, and Q_{max} is set to zero. For our algorithm, we set the number of tasks in phase 1 to 15 according to Lemma 2 with uniform task sampling and $\delta = 0.02$. The Q-gap Γ is set to 2. The number of trajectories we run in phase 2 to approximate the Q value, T_i, is set to 1 because the domain is deterministic. For PG-ELLA, the learning rate is selected automatically in their code.

Figure 1 shows the average reward for each of the 50 tasks for the 4 algorithms. As expected, our approach performs comparably to single task C-PACE in phase 1, with a slight loss in performance because we try to fully explore the state–action space. However our algorithm significantly improves in phase 2, successfully accelerating learning by identifying tasks quickly. In phase 2, the gap between our method and single-task C-PACE implies the superiority of transferring samples from previous tasks, which also confirms the analysis of sample complexity in an empirical way. For single task C-PACE could not collect enough samples to converge

[4]For computational efficiency and to focus on the key impact of transferring knowledge from multiple tasks, we chose this discrete action space. Alternatively we could use the Lipschitz constant to define a discretization.

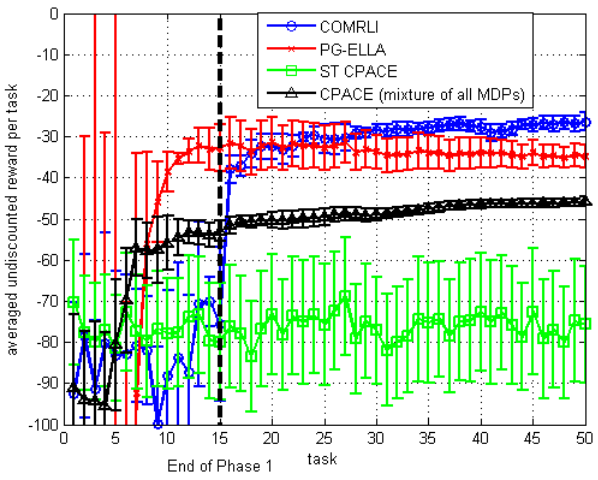

Figure 1: Averaged reward per task. The bar is standard deviation over 40 rounds.

in just one task, its standard deviation is quite big as, as it was shown in the figure. Our algorithm is also better than the C-PACE algorithm that treats all tasks the same, which shows the benefit of learning a separate optimal policy per distinct task instead of a single policy for all tasks.

During phase 2 our algorithm exceeds the performance of PG-ELLA. Note that PG-ELLA uses the first several tasks to learn a shared basis, which we show in the figure, but is omitted in the graphs of their paper. We consider this quite encouraging, since PG-ELLA is provided the identity of each task, compared to our approach which does not have that information. Of course, our algorithm is operating in a domain in which tasks have Q functions that are quite well separated, as the Q-gap assumption.

These preliminary results suggest that our approach can perform well in standard benchmark simulation, with a significant advantage over single task algorithms and naive transfer, and equivalent or slightly improved performance over a state-of-the-art method, PG-ELLA. As our approach has rigorous guarantees on sample complexity, which PG-ELLA lacks, these empirical results are quite encouraging.

7. CONCLUSION AND FUTURE WORK

We have introduced the COMRLI algorithm, a PAC RL algorithm for learning across a series of tasks drawn from a set of continuous-state, discrete action Markov decision processes. To our knowledge, COMRLI is the first algorithm for multi-task learning in continuous state RL problems whose bound on the sample complexity can be significantly smaller in later tasks, and is independent of the covering number for the state–action space. This shows that transferring knowledge in continuous-state RL can provably reduce the amount of experience needed to make good decisions. We also provide preliminary but encouraging evidence that our approach may be helpful in practice, showing good performance in one benchmark domain against a state-of-the-art policy gradient method for multi-task RL.

8. ACKNOWLEDGEMENTS

This work was supported by NSF CAREER award #1350984.

REFERENCES

[1] H. B. Ammar, E. Eaton, P. Ruvolo, and M. Taylor. Online multi-task learning for policy gradient methods. In T. Jebara and E. P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1206–1214. JMLR Workshop and Conference Proceedings, 2014.

[2] H. B. Ammar, R. Tutunov, and E. Eaton. Safe policy search for lifelong reinforcement learning with sublinear regret. *The Journal of Machine Learning Research (JMLR)*, 2015.

[3] P. Auer, T. Jaksch, and R. Ortner. Near-optimal regret bounds for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 89–96, 2009.

[4] H. Bou-Ammar, J. M. Luna, E. Eaton, and P. Ruvolo. Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[5] E. Brunskill and L. Li. Sample complexity of multi-task reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*, 2013.

[6] R. Grande, T. Walsh, and J. How. Sample efficient reinforcement learning with gaussian processes. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1332–1340, 2014.

[7] R. Grande, T. Walsh, and J. How. Sample efficient reinforcement learning with gaussian processes. In T. Jebara and E. P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1332–1340. JMLR Workshop and Conference Proceedings, 2014.

[8] Z. Guo and E. Brunskill. Concurrent pac rl. In *AAAI Conference on Artificial Intelligence*, 2015.

[9] S. Kakade, M. Kearns, and J. Langford. Exploration in metric state spaces. In *ICML*, volume 3, pages 306–312, 2003.

[10] S. M. Kakade et al. *On the sample complexity of reinforcement learning*. PhD thesis, University of London, 2003.

[11] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

[12] A. Lazaric and M. Restelli. Transfer from multiple mdps. In *Advances in Neural Information Processing Systems*, pages 1746–1754, 2011.

[13] R. Liu and K. R. Koedinger. Variations in learning rate: Student classification based on systematic residual error patterns across practice opportunities.

[14] T. A. Mann and Y. Choe. Directed exploration in reinforcement learning with transferred knowledge. In *EWRL*, pages 59–76, 2012.

[15] S. Nikolaidis, R. Ramakrishnan, K. Gu, and J. Shah. Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 189–196. ACM, 2015.

[16] J. Pazis and R. Parr. Pac optimal exploration in continuous space markov decision processes. In *AAAI Conference on Artificial Intelligence*. Citeseer, 2013.

[17] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, 10:1633–1685, 2009.